10

Para aprender más: Cuidado con el modelo anémico

Durante el aprendizaje de Getters y Setters es normal pensar siempre en su necesidad de cambiar algún estado de nuestros objetos.

Pero el uso de esta práctica no siempre es el más adecuado y expresa la realidad.

Tenga en cuenta la clase `Cuenta` representada a continuación que usa solo getter y setters como métodos:

```
class Cuenta{
    private String titular;
    private double saldo;

    public void setTitular(String titular){
        this.titular = titular;
    }

    public String getTitular(){
        return titular;
    }

    public void setSaldo(double saldo){
        this.saldo = saldo;
    }

    public double getSaldo(){
        return saldo;
    }
```

```
}
```

COPIA EL CÓDIGO

Continuamos usando atributos privados y nuestro modelo parece seguir perfectamente la propuesta de encapsulación donde la clase misma administra sus estados (atributos). Un uso clásico de esa cuenta nos llevaría al siguiente escenario:

```
Cuenta cuenta = new Cuenta();
cuenta.setTitular("Fábio")
cuenta.setSaldo(100.0);
```

COPIA EL CÓDIGO

Todo parece perfecto, ahora imagina que necesitas retirar 50.0 de esa cuenta. Esta operación requerirá que el saldo sea suficiente. Una simple verificación de la siguiente manera aseguraría que el saldo no haya sido negativo. En nuestro ejemplo no hay límite más allá del saldo :)

```
Cuenta cuenta = new Cuenta ();
cuenta.setTitular("Fábio")
cuenta.setSaldo(100.0);

double valorSaque = 50.0

if(cuenta.getSaldo() >= valorSaque){
    double nuevoSaldo = cuenta.getSaldo() - valorSaque;
    cuenta.setSaldo(nuevoSaldo)
}
```

¡Funcionó! Pero un problema es que esta lógica de restringir e

```
Cuenta conta = new Cuenta ();
cuenta.setTitular("Fábio");
cuenta.setSaldo(100.0);

double valorSaque = 50.0;

if(cuenta.getSaldo() + 1000.0 >= valorSaque){
    double nuevoSaldo = conta.getSaldo() - valorSaque;
    cuenta.setSaldo(nuevoSaldo)
}
```

Cuando creamos clases que se limitan a tener atributos privado

¡Una clase de títeres es una que no tiene responsabilidad, apa

Volviendo a nuestro ejemplo de la Cuenta, está claro que en el

COPIA EL CÓDIGO

class Cuenta{ private String titular; private double saldo;

```
public void setTitular(String titular){
    this.titular = titular;
}

public String getTitular(){
    return titular;
}
```

```
    public void saca(double valor){
        if(valor > 0 && saldo >= valor){
            saldo -= valor;
        }
    }

    public void deposita(double valor){
        if(valor>0){
            saldo += valor;
        }
    }

    public double getSaldo(){
        return saldo;
    }
```

COPIA EL CÓDIGO

```
}
```

```
    Tenga en cuenta que la lógica de saque y depósito está represe
```

COPIA EL CÓDIGO

Cuenta cuenta = new Cuenta(); cuenta.setTitular("Fábio");
cuenta.deposita(100.0);

double valorSaque = 50.0; cuenta.saca(valorSaque);

`

¿Mucho mejor no es así? No hay duplicaciones de código, y mucho menos otras clases que controlan el estado de nuestra cuenta como lo hicimos anteriormente.

Conclusión Los setters y getters deben usarse con precaución y no todos los atributos privados necesitan exponer estos dos métodos en riesgo de caer en un `modelo anémico` que tiene sus comportamientos controlados por otras clases.